Using categorical embeddings (deep learning) in boosting models

WHY-R? CONFERENCE
29. 09. 2019
WARSAW, POLAND
TAMÁS BURGHARD

Who Am I?

- Data scientist @ ingatlan.com
  (real estate listing website in Hungary)
  (thanks for the support!)
- Studied Business Analitics @ CEU
- You can reach me at:

tamas.burghard@gmail.com

WHY-R U here?

- This talk tries to be a thought-provoking one - urge you to utilize other disciplines' ideas and results in data science ; **transfer learning in different meanings**
- Warning: some of the methods shown have no (mature) library in R

Topics: (slightly altered from the original)

- Definition of high cardinality categorical variable and implications on Xgboost training
- Understanding embedding vectors
- Different embedding methods

Similarity/dissimilarity and distance

- These methods emerged from disciplines caring with similarity-dissimilarity. Recommender systems imply that similar users like similar items. In NLP, next word after a specific one represent underlying structure.
- Similar things are closer to each other in a structure – you have one kind of distance.
- Famous NLP example with word vectors: adding ‘Woman’ to ‘King’ and subtracting ‘Man’ gave ‘Queen’.

Categorical variable with high cardinality and boosting trees

- A categorical variable (factor) with more than 10 levels (or 5; 50) – these tend not to be ordered categories
- One must encode the variable into numbers for machine learning
- Boosting trees: the ideal cut creates 2 dissimilar parts that are similar inside (regarding the outcome – like it was ordered by that). The perfect order changes as the algo cuts the space by other variables. **Different orders for the same variable can be useful! (and a quick way to overfit)**
- Using factors in R (or integer encoding)
  Has one order only – you can tweak this with mean encoding (overfit). Most probably that order is not good at all. Distance and similarity unknown between levels.

Encoding with vectors instead of a scalar

- More variables encode the same info = more orders you possibly have.
- One hot : can choose one level - or all other with one split. Creates too much new vars, too much cuts needed. If you do scaling for anything needs to be weighted, you should consider dividing with the cardinality. Also affects the column sampling part of Xgboost – you must find new optimal hyperparameters. Choose one-hot, not dummy. Distance is problematic.
- Limiting the number of new vars seems reasonable. PCA is for continuous var, NOT bools. Lasso loosing levels. Can try MCA or similar.
- Embedding vector (of k dim): k new continuous(numeric) columns. The basic idea behind: similar objects in the original space will get similar vectors. One hot encoding is an embedding vector ! (just not a good one)

Example: Date as an embedding

- Think of selling electronic devices online. Products sold is the outcome, date is one predictor. You might try to ease your algorithm with having ‘year’, ‘quarter’, ‘month’, ‘day of month’, ‘is_weekend’, ‘is_holiday’, ‘is_black_Friday’ as features.
  A date embedding could represent this with less variables. It’s similar to genes: some position in the vector might encode a single property, but mostly several positions collaborate with each other.
- The vector represents the underlying data structure, through similarity. Therefore it is useful when similarity or distance is important.

3 ways to get embedding vectors

- Creating embedding vectors for every structure is very popular nowadays. They can compact information about housing ads, location, programming code snippets, etc.

I would like to introduce you 3 examples

- Embedding layer of deep learning (no explicit similarity)
- Matrix factorization (in recommender systems)
- Ads on Airbnb, through user session data (Specialized techniques for sequences with **custom loss** functions - anything to vec)

Emdedding vectors from deep learning: step_embed from the embed package

- Part of the tidymodels ecosystem
- Transfer learning: different domain, same dataset, possibly different task.
- At the background, this is a lookup table from the levels of the cat.vat. To k dimension vectors. A shallow net tries to solve a problem using this as an input, and can adjust the values of the lookup according the error.
- Needs Keras to be installed (a Deep Learning library)
- Better to encode more categorical variables in one step – but they will have the same dimension
- Use numeric features, normalized! Set hidden to 10 or 100.
- You can set verbose=1 to see what's happening. Nan in loss implies division by zero (scale with 1 level?), or having NA somewhere.
- The dataset used is from Szilárd Pafka's GBM performance page.

Sample code

```
 step_embed(Origin, Dest, UniqueCarrier,          # the 3 cols to encode
            outcome = vars(dep_delayed_15min_Y), # outcome for THIS step
            predictors = vars(Distance, DepTime),# 2 already numeric var
            num_terms = 10L,                     # dimension of the emb
            hidden_units = 10L,
            options = embed_control(verbose = 1, validation_split = .2,
                                    batch_size = 64, epochs = 16)
  )
```

For the full script please visit: https://github.com/burgikukac/why-r_2019

Results

All trained with 1m train, 100 rounds, 0.1 eta, 10 max.depth		
	AUC	Nr. Of vars
Integer encoding (reference)	0.745869	9 vars
One-hot	0.740694	652 vars
3 embedding	**0.753045**	35 vars
All cat embedded	0.749953	63 vars
Nb. All of the models were underfit		

Collaborative filtering and matrix factorization

- Idea behind: similar users like similar items.
- One big (and sparse) matrix: rows users, columns items, elements a score. Most of the (user, item) pairs have no score. This matrix can be represented as a product of a User mtx, and an Item mtx -> if both users and items have the same dimension k: (u, k)*(k,i) (not exact representation) Basically score is just a dot product of the vector of the specific user and the specific item.
- You can use this for any 2 categories and your target. The matrix above were the wide format of this. I use this for filling NA.
- Watch Jeremy's explanation on fast.ai.

Embedding and collaborative filtering

- Almost the same code in Keras.
- If you need custom embedding, you can always train a deep net by hand. Embedding layer is available for you.
- Embedding with embed is like concatenating the vectors together.
- Embedding with matrix fact. Is the dot product of the 2 embedding vectors. (+ biases)

Airbnb example

- Different domain, different datasets
- Airbnb has info about the ads (and the properties themselves) , and also the user activities (searches and clicks per session)
- Two properties/ads are similar, if they located in the same zone (or were in the domain of the search), and the user clicked on them. Those were not seen but could have (were in the search hits), are dissimilar. (negative sampling)
- Technically a neural net, input as embedding layer, the magic is in the loss function and the selection of the click data.
- I am not aware of a library in R doing this translate from sequences to vectors, only the original NLP idea. Could be interesting, as tons of companies have online data.

Sources, further readings

- Cheng Guo, Felix Berkhahn, ” Entity Embeddings of Categorical Variables’ arXiv preprint arXiv:**1604.06737 (2016)**
- Daniele Micci-Barreca. 2001. **A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems**. SIGKDD Explor. Newsl. 3, 1 (July 2001), 27-32.
- Mihajlo Grbovic, Nemanja Djuric, Vladan Radosavljevic, Fabrizio Silvestri, Ricardo Baeza-Yates, Andrew Feng, Erik Ordentlich, Lee Yang, and Gavin Owens. 2016. **Scalable Semantic Matching of Queries to Ads in Sponsored Search Advertising**. In Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval (SIGIR '16). ACM, New York, NY, USA, 375-384. **arXiv:1607.01869**
- Mihajlo Grbovic and Haibin Cheng. 2018. **Real-time Personalization using Embeddings for Search Ranking at Airbnb**. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '18). ACM, New York, NY, USA, 311-320. https://astro.temple.edu/~tua95067/kdd2018.pdf
- Embeddings, collaborative filtering : Fast.ai course https://course.fast.ai/videos/?lesson=4 (the course is python-based, but this part is described very well – there is also an Excel spreadsheet doing the very same thing!
- Mean encoding and other useful methods: How to Win a Data Science Competition: Learn from Top Kagglers (Coursera)
- https://github.com/tidymodels
- https://github.com/szilard/GBM-perf

Thank you for your attention!